Machine Learning – An Introduction

Prof. Nicholas Zabaras
Center for Informatics and Computational Science
https://cics.nd.edu/
University of Notre Dame
Notre Dame, Indiana, USA

Email: nzabaras@gmail.com
URL: https://www.zabaras.com/

January 15, 2019

Course Web Site: https://www.zabaras.com/machine-learning

Contents

- Machine learning, Types of machine learning
- Supervised Learning, Unsupervised Learning, Reinforcement Learning
- Supervised learning: Classification, Probabilistic Predictions, Point Estimates, Document Classification, Iris Flower Dataset, Image Classification, Face Recognition and detection, Regression, Unsupervised Vs. Supervised Learning
- Unsupervised learning, Hidden/Latent Variables, Dimensionality Reduction, Discovering Graph Structure, Matrix Completion
- Parametric vs non-parametric models, K -nearest neighbors
- The curse of dimensionality
- Inductive bias, Linear regression, Logistic regression, Overfitting, Model selection, Cross Validation
- No free lunch theorem

- Kevin Murphy's, Machine Learning: A probablistic perspective, Chapter 1

Machine Learning

- We are in the era of *big data* (size of the web, youtube, etc.)
- Automated methods of data analysis: *detect patterns in the data, predict future data, decision making with uncertainty, etc.*
- The probabilistic approach to machine learning is closely related to the field of computational statistics.

- Rajaraman, A. and J. Ullman (2010). *Mining of massive datasets*. To appear
- Bekkerman, R., M. Bilenko, and J. Langford (Eds.) (2011). *Scaling Up Machine Learning*. Cambridge (online presentation)

Supervised Learning

- Machine learning is divided into two types. In the *supervised learning approach*, the goal is to learn a mapping from inputs $\boldsymbol{x}$ to outputs y, *given a labeled set of input-output pairs* $\mathcal{D} = (\boldsymbol{x}_i, y_i), i = 1, .., N$. $\mathcal{D}$ is called the **training set**, and N is the number of training examples.

- $\boldsymbol{x}_i$ is a D-dimensional vector of *features, attributes or covariates*.

- The form of the output (response variable) can in principle be anything. Most methods assume that y_i is a categorical $y_i = 1, \ldots, C$ (e.g. male or female), or that y_i is a real-valued scalar (e.g. income level).
  - When y_i *is categorical, the problem is known as classification or pattern recognition*
  - When y_i *is real-valued, the problem is known as regression*
  - In *ordinal regression* the response y has some natural ordering (e.g. grades A-F)

Unsupervised Learning

- In *unsupervised learning approach,* we are given input data $\mathcal{D} = (\boldsymbol{x}_i), i = 1, .., N$ and the goal is to find "interesting patterns" in the data (*knowledge discovery*).

- *We are not told what kinds of patterns to look for, and there is no obvious error metric to use*

  - ✓ *This is* unlike supervised learning, where we can compare our prediction of y for a given $\boldsymbol{x}$ to the observed value.

Reinforcement Learning

- There is a third type of machine learning, known as ***reinforcement learning***.
- This is for *learning how to act or behave given occasional reward or punishment signals*.
- We discuss next typical examples of supervised & unsupervised learning.

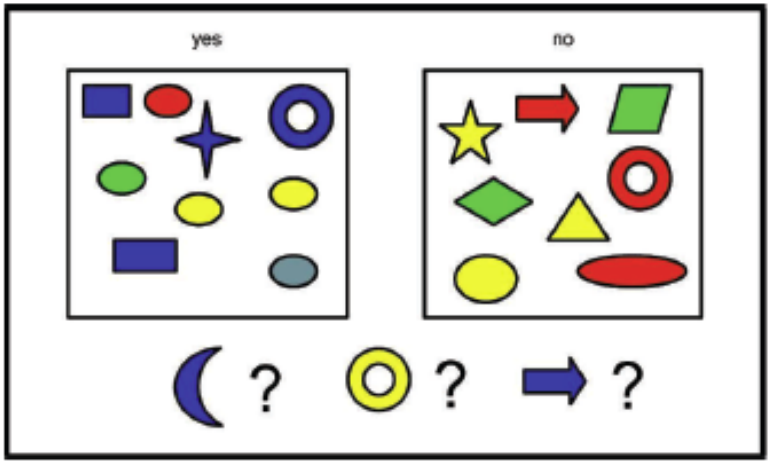
- Kaelbling, L., M. Littman, and A. Moore (1996). Reinforcement learning: A survey. *J. of AI Research 4*, 237–285.
- Sutton, R. and A. Barto (1998). *Reinforcment Learning: An Introduction*. MIT Press
- Russell, S. and P. Norvig (1995). *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall.
- Szepesvari, C. (2010). *Algorithms for Reinforcement Learning*. Morgan Claypool.
- Wiering, M. and M. van Otterlo (Eds.) (2012). *Reinforcement learning: State-of-the-art*. Springer.

Supervised Learning: Classification

- We learn a mapping from inputs $\boldsymbol{x}$ to outputs y, where $y_i \in \{1, 2, ..., C\}$, with C being the number of classes.
  - $C = 2$, *binary classification*.
  - $C > 3$, *multiclass classification*.
- If the class labels are not mutually exclusive (e.g., somebody may be classified as tall and strong), we call it *multi-label classification*.
- Our interest is on *generalization* – making predictions on novel inputs.

Supervised Learning: Classification

- Left: Training examples of colored shapes, along with 3 unlabeled test cases.
- Right: Training data as an $N \times D$ *design matrix*. Row i represents the feature vector x_i. The last column is the label, $y_i \in \{0,1\}$



D features (attributes)

N cases

Color	Shape	Size (cm)	Label
Blue	Square	10	1
Red	Ellipse	2.4	1
Red	Ellipse	20.7	0

Probabilistic Predictions

- In our classification example, we work with posterior probabilities $p(y|\boldsymbol{x}, \mathcal{D})$, $\boldsymbol{x}$ is the input vector and $\mathcal{D}$ is the training set. E.g. for binary classification $y = 1$ or $y = 0$.
- Given a probabilistic output, we can always compute our "best guess" as to the "true label" using

$$\hat{y} = \hat{f}(\boldsymbol{x}) = \underset{c=1,..,C}{\operatorname{argmax}} p(y = c|\boldsymbol{x}, \mathcal{D})$$

- This corresponds to the *most probable class label*, and is called the mode of the distribution $p(y|\boldsymbol{x}, \mathcal{D})$. It is also known as a *MAP estimate* (*maximum a posteriori*).
- Point estimates are often not the best solution -- how about if $p(\hat{y} = 1|\boldsymbol{x}, \mathcal{D})$ is far from 1 (e.g. yellow circle on our previous test set)?
- We need *confidence on our predictions.*

Point Estimates

- Point estimates can be misleading and need to be avoided.
- This is important in medicine and finance where we may be risk averse.
- IBM Watson beat the top human Jeopardy champion by containing a module that estimates how confident it is of its answer.
- Google's SmartASS (ad selection system) predicts the probability (click-through rate, CTR) you will click on an ad based on your search history and other user and ad-specific features. CTR can be used to maximize expected profit.

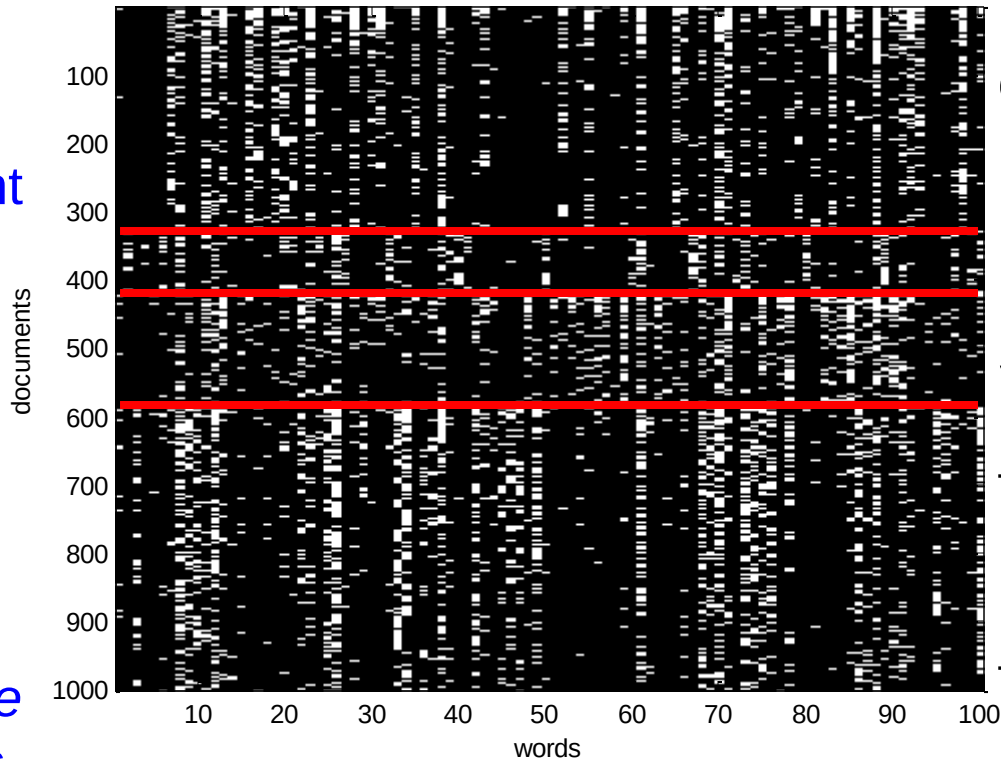
- Ferrucci, D., E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefter, and C. Welty (2010). Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 59–79.
- Metz, C. (2010). Google behavioral ad targeter is a Smart Ass. *The Register*.

Document Classification and Email Filtering

- *We use a bag of words representation* ($x_{ij} = 1$ if word j appears in document i). This leads to a binary document $\times$ (times) a word co-occurrence matrix.

- A subset of size 16242×100 of the $20-$newsgroups data. Each row is a document (bag-of-words bit vector), *each column is a word*.

- The red lines separate the 4 classes. We can see that *there are subsets of words whose presence or absence is indicative of the class*.



newsgroupsVisualize from PMTK

Classifying Flowers

- Here the goal is to learn to distinguish 3 types of iris flower. *Rather than working directly with images, 4 features have been extracted.*



fisheririsDemo from PMTK

- One can learn good features from the data.
- From a ***scatter plot***, we can distinguish setosas (red circles) from the other two classes by checking if their petal length/ width is below a threshold.
- Distinguishing other types is harder and needs to be based on at least two features (perform ***exploratory data analysis*** – *e..g plotting the data - before* applying a machine learning method)

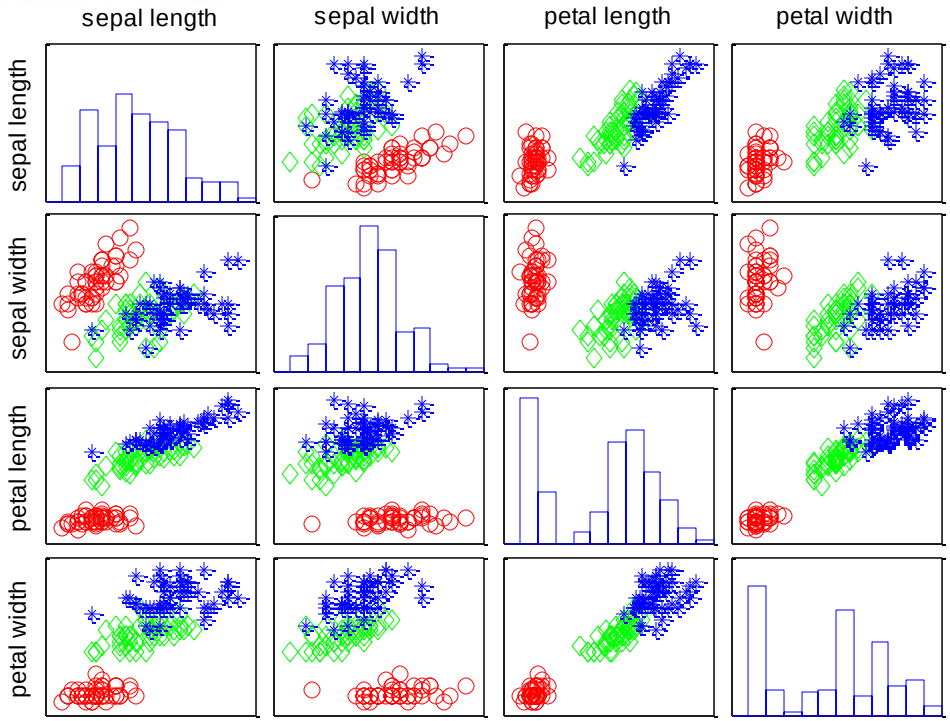
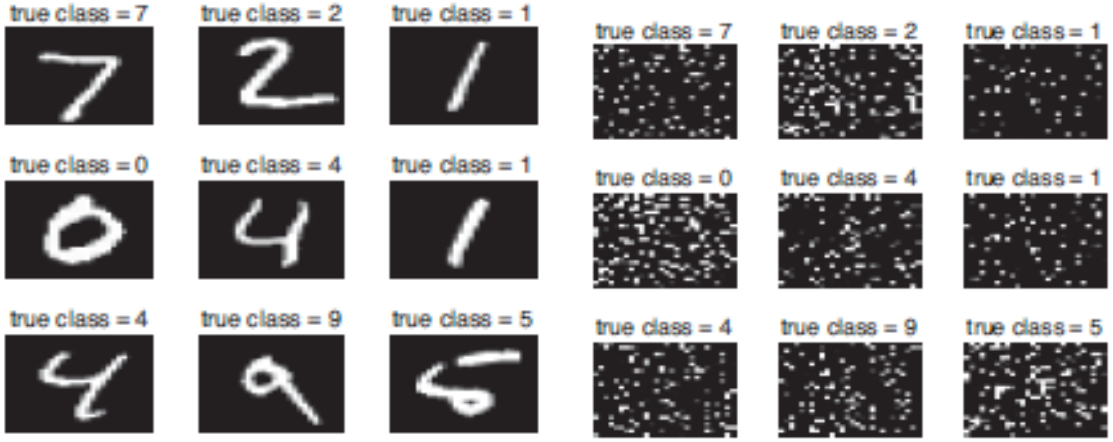


Image Classification & Handwriting Recognition

- Consider ***the problem of classifying images*** **directly with no preprocessing**. We want to classify the image as a whole, e.g., is it an indoors or outdoors scene?



Run shuffledDigitsDemo from PMTK

- In ***handwriting recognition*** (dataset MNIST), the images are size 28×28 and have grayscale values in the range $0:255$.
- Most generic classification methods ignore any structure in the input features including spatial layout.
  - ✓ They can as easily handle data that looks like on the Fig. on the right (same data but with randomly permuted order of features).
- General methods are important but ignore useful source of information.

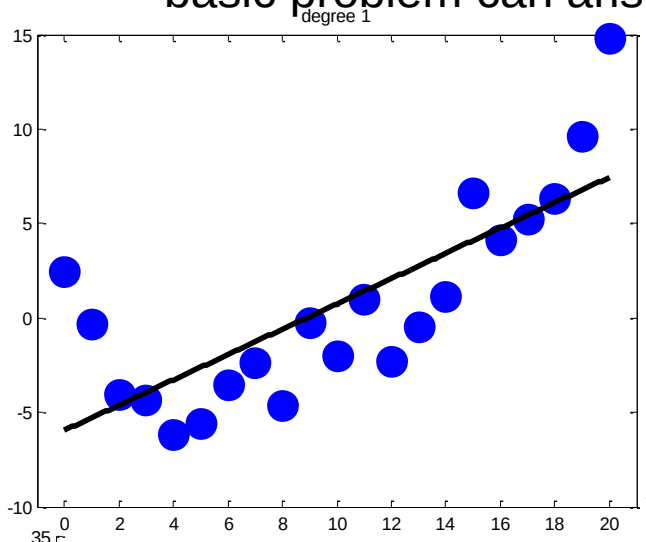
Face Recognition

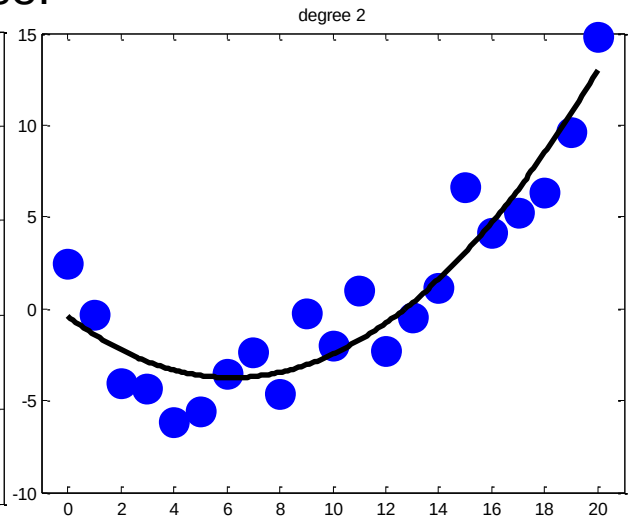
- A harder problem is ***object detection/localization***. ***Face detection*** is a special case.
- Divide the image into many small overlapping patches at different locations, scales & orientations, and classify each patch based on whether it contains face-like texture or not ***(sliding window detector)***.
- The system returns those locations where the probability of face is high. Such systems are built-in the *auto-focus of digital cameras*.
- Another application is blurring out faces in *Google's StreetView system*.
- Having found the faces, one can then proceed to perform ***face recognition***. In this case, the number of class labels is large and the features to use more subtle (e.g. hairstyle) and different than in the face detection problem.
- *Detection is **invariant** to face details – we are only interested in differences between faces & non-faces.*

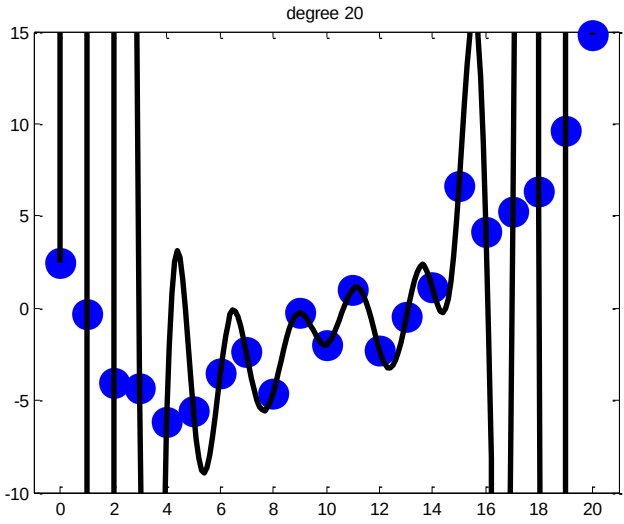
- Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer

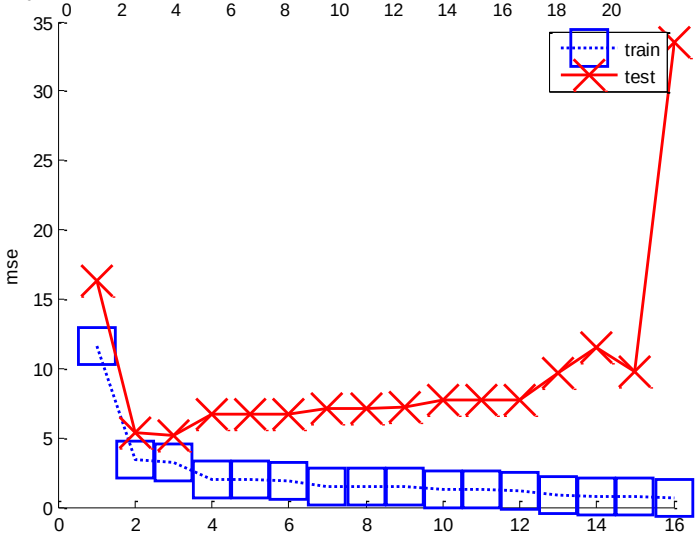
Supervised Learning: Regression

- Consider a real-valued input $x_i \in \mathbb{R}$, and a single real-valued response $y_i \in \mathbb{R}$. We fit a straight line and a quadratic function. Various extensions of this basic problem can arise.









- Many applications with high dimensional input data.
- Issues of model selection are essential (overfitting, etc.)

linRegPolyVsDegree from PMTK

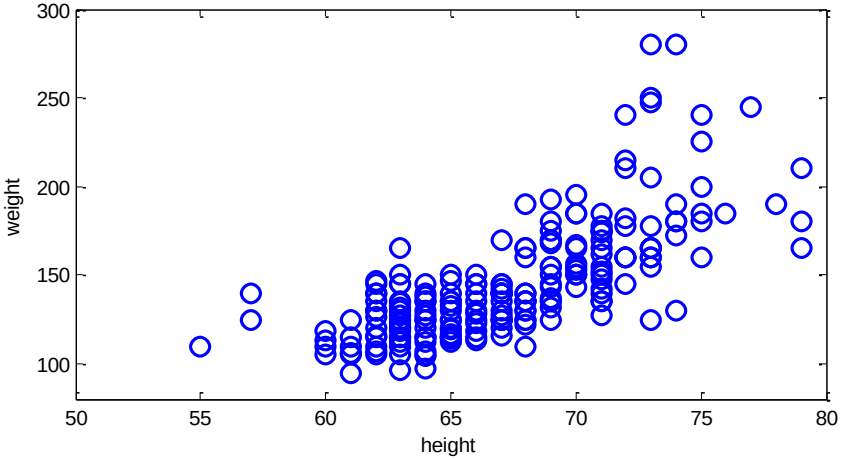
Unsupervised Vs. Supervised Learning

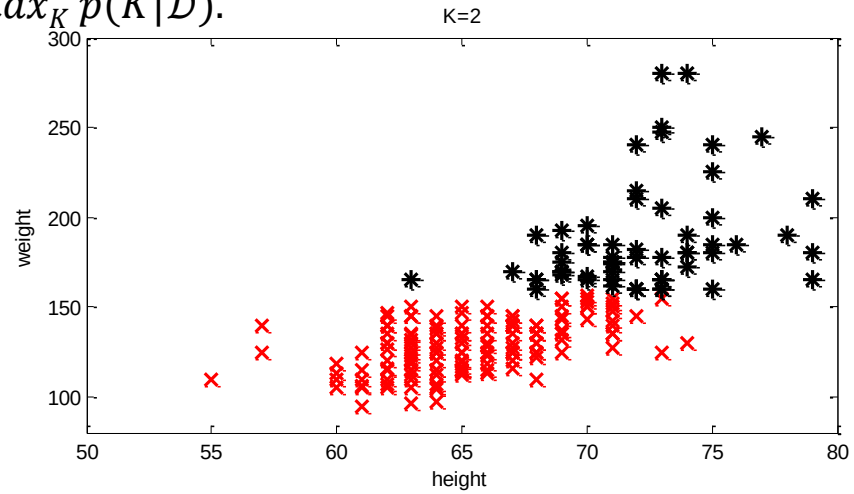
- We are *just given output data, without any inputs*. The goal is to *discover structure in the data (**knowledge discovery**)*.
- Unlike supervised learning, we don't know the desired output for each input. This is a ***density estimation problem***, i.e. *build* $p(\boldsymbol{x}_i|\boldsymbol{\theta})$.
- There are two differences from the supervised case.
  - Supervised learning is conditional density estimation, $p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta})$.
    - ✓ y_i is usually a single variable (class label) we are trying to predict. Thus for most supervised learning problems, we can use univariate probability models.
  - *Unsupervised learning is unconditional density estimation*, $p(\boldsymbol{x}_i|\boldsymbol{\theta})$.
    - ✓ *$\boldsymbol{x}_i$ is a vector of features, so we need to create multivariate probability models*.

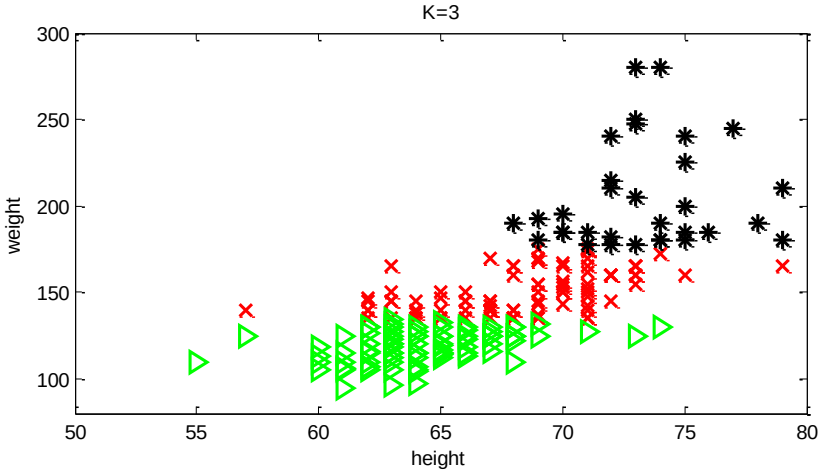
- Cheeseman, P., J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman (1988). Autoclass: A Bayesian classification system. In *Proc. of the Fifth Intl. Workshop on Machine Learning*.
- Lo, C. H. (2009). *Statistical methods for high throughput genomics*. Ph.D. thesis, UBC.
- Berkhin, P. (2006). A survey of clustering datamining techniques. In J. Kogan, C. Nicholas, and M. Teboulle (Eds.), *Grouping Multidimensional Data: Recent Advances in Clustering*, pp. 25–71. Springer.

Unsupervised Learning: Hidden Variables

- Consider **clustering** data into groups -- height and weight of a group of 210 people. It is not clear how many clusters we have.
- Our first goal is to estimate the distribution over the number of clusters, $p(K|\mathcal{D})$; this tells us if there are subpopulations within the data. For simplicity, we often approximate the distribution $p(K|\mathcal{D})$ by its mode, $K^* = \arg max_K \, p(K|\mathcal{D})$.







- The second objective is to assign each data point to the corresponding cluster (*hidden or latent variables*).

$$z_i^* = \arg\max_k \, p(z_i = k \mid \mathbf{x}_i, \boldsymbol{\mathcal{D}})$$

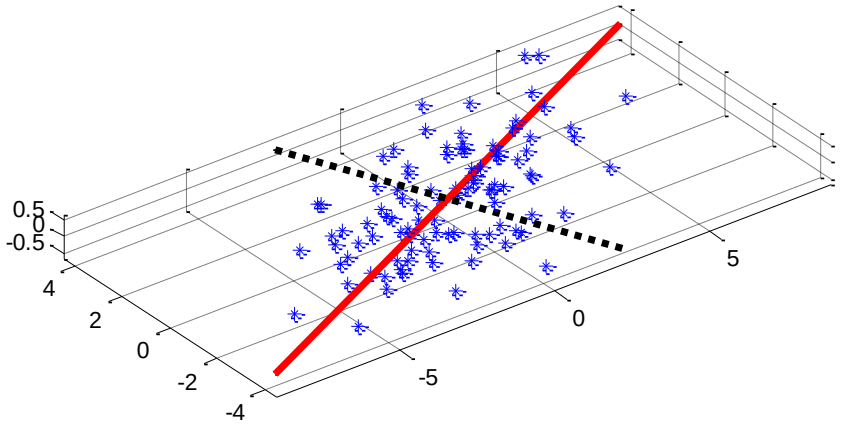
- Picking a model of the right complexity (here the number of clusters) is called *model selection*.

kmeansHeightWeight from PMTK

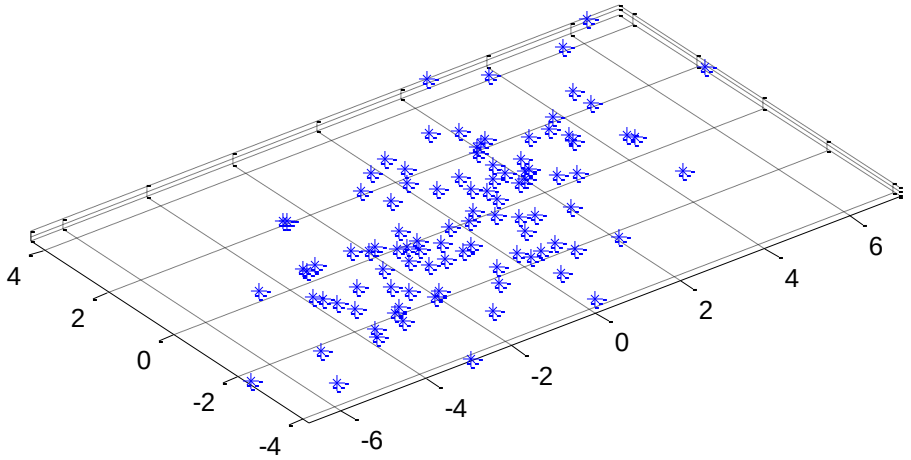
Dimensionality Reduction

- Reduce the dimensionality by projecting the data to a lower-dimensional subspace which captures the essence of the data.
- *Latent factors:* although the data may appear high-dimensional, there may only be a small number of degrees of variability.
- *Principal Components Analysis (PCA):* common approach to dimensionality reduction. Useful for visualization, nearest neighbor searchers, etc.

pcaDemo3d from PMTK

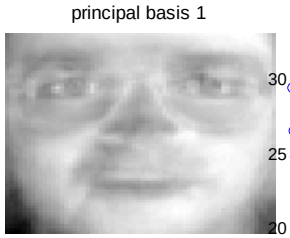


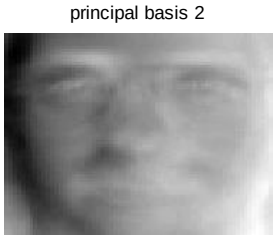
A set of points that live on a 2d linear subspace embedded in 3d. The 2 ``principal directions'' are shown.

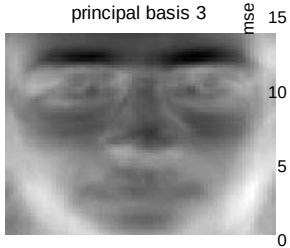


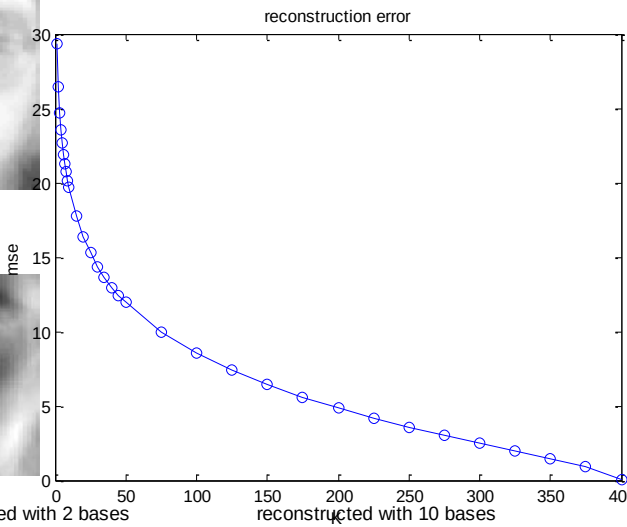
2D representation of the data

Dimensionality Reduction









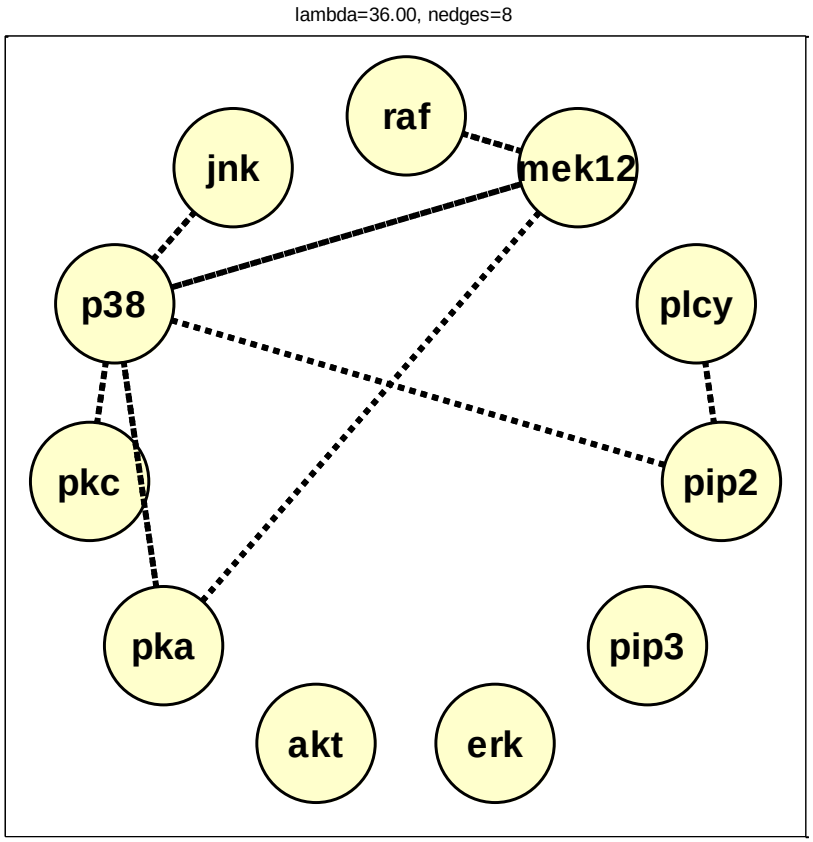
- PCA can be thought of as an unsupervised version of (multi-output) linear regression.
  - ✓ *Actual generative model $z \rightarrow y$*
  - ✓ *Here we go the other way $y \rightarrow z$, i.e. infer the latent low-dimensional $\boldsymbol{z}$ from the observed high-dimensional $\boldsymbol{y}$.*

Run MatLab function
pcaImageDemo from Kevin Murphys' PMTK

Discovering Graph Structure

- We measure a set of variables, and we like to discover which ones are most correlated with which others. This is represented by a graph G, in which nodes represent variables, and edges represent dependence between variables. We look to compute: $\hat{G} = \text{argmax} p(G|\mathcal{D})$

A sparse undirected Gaussian graphical model is shown learned using *graphical lasso* applied to some flow cytometry data which measures the phosphorylation status of 11 proteins.



ggmLassoDemo from PMTK

- Sachs, K., O. Perez, D. Pe'er, D. Lauffenburger, and G. Nolan (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science 308*.
- Smith, V., J. Yu, T. Smulders, A. Hartemink, and E. Jarvis (2006). Computational Inference of Neural Information Flow Networks. *PLOS Computational Biology 2*, 1436–1439
- Horvitz, E., J. Apacible, R. Sarin, and L. Liao (2005). Prediction, Expectation, and Surprise: Methods, Designs, and Study of a Deployed Traffic Forecasting Service. In *UAI*.
- Carvalho, C. M. and M. West (2007). *Bayesian Analysis* *2*(1), 69–98.

Matrix Completion

- Sometimes we have missing data. The goal of imputation is to infer plausible values for the missing entries. This is sometimes called matrix completion.

- Below we give some example applications.
  - Image inpainting
  - Collaborative filtering (e.g. movie rating data)
  - Market basket analysis (e.g. purchasing patterns)

- Hu, D., L. van der Maaten, Y. Cho, L. Saul, and S. Lerner (2010). Latent Variable Models for Predicting File Dependencies in Large-Scale Software Development. In *NIPS*.

Parametric Vs. Nonparametric Modeling

- *Parametric model, the model have a fixed number of parameters.* Parametric models have the advantage of often being faster to use, but the disadvantage of making stronger assumptions about the nature of the data distributions.
- *Non parametric model, the number of parameters grow with the amount of training data.* Non parametric models are more flexible, but often computationally intractable for large datasets.

K-Nearest Neighbor Classifier

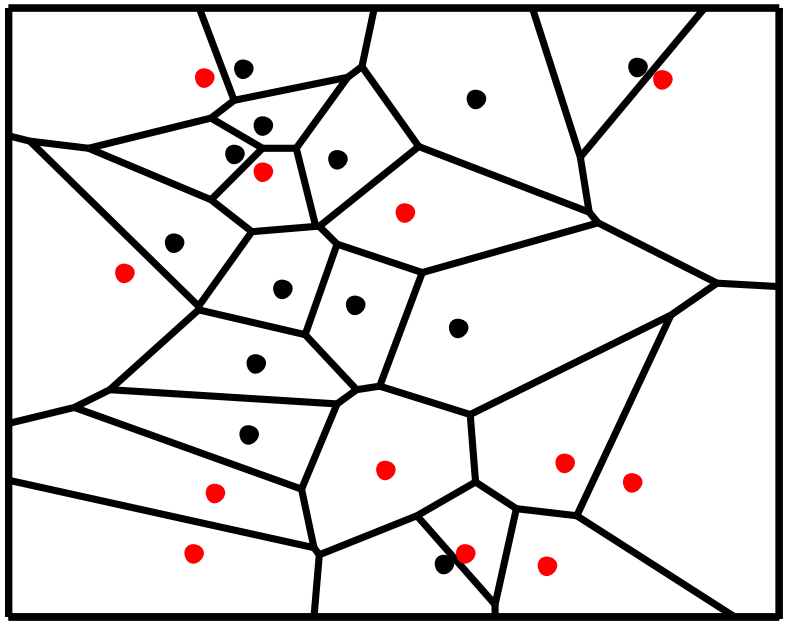
- A simple example of a non-parametric classifier is the K nearest neighbor (KNN) classier.

- This simply *looks at the K points in the training set that are nearest to the test input $\boldsymbol{x}$, counts how many members of each class are in this set, and returns that empirical fraction as the estimate*

$$p(y = c|\boldsymbol{x}, \mathcal{D}, K) = \frac{1}{K} \sum_{i \in N_k(\boldsymbol{x}, \mathcal{D})} \mathbb{I}(y_i = c)$$

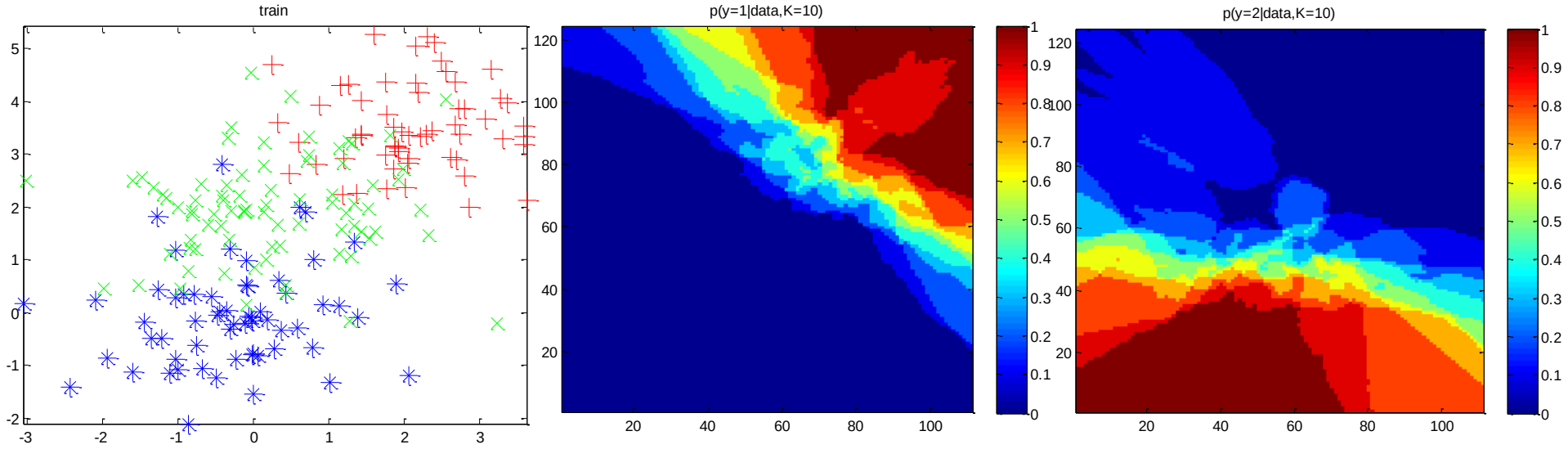
where $N_k(\boldsymbol{x}, \mathcal{D})$ are the (indices of the) K nearest points to $\boldsymbol{x}$ in $\mathcal{D}$.

- Using $K = 1$ leads to a *Voronoi Tessellation* – *all points in $V(\boldsymbol{x}_i)$ are closer to $\boldsymbol{x}_i$ than to any other point.*

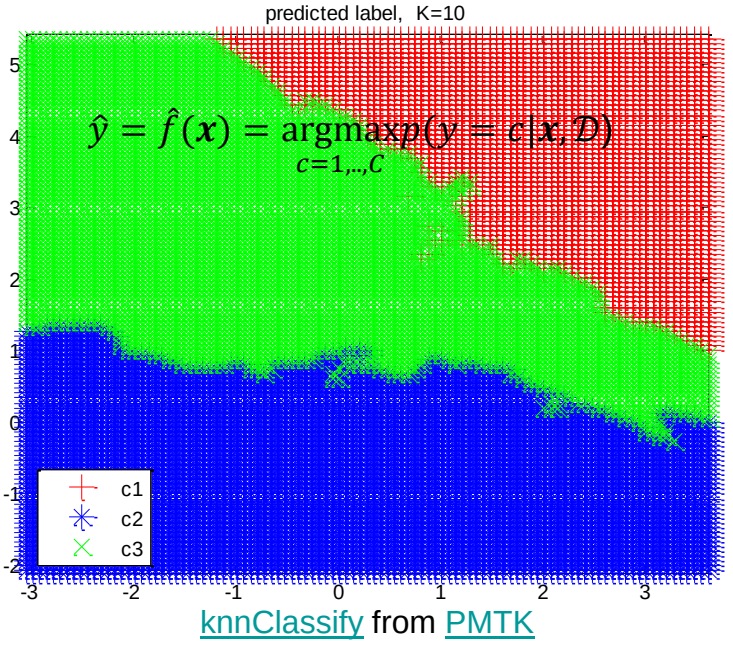


knnVoronoi from PMTK

K-Nearest Neighbor Classifier



(a) Synthetic 3-class training data in $2d$. (b) Probability of class 1 for KNN with $K = 10$. (c) Probability of class 2. (d) MAP estimate of class label.



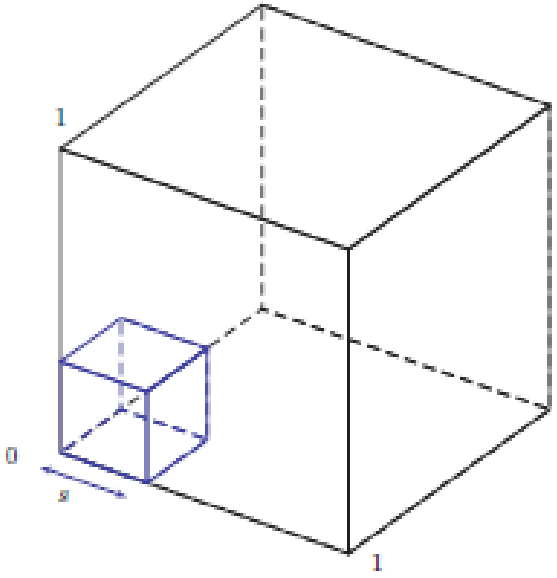
knnClassify from PMTK

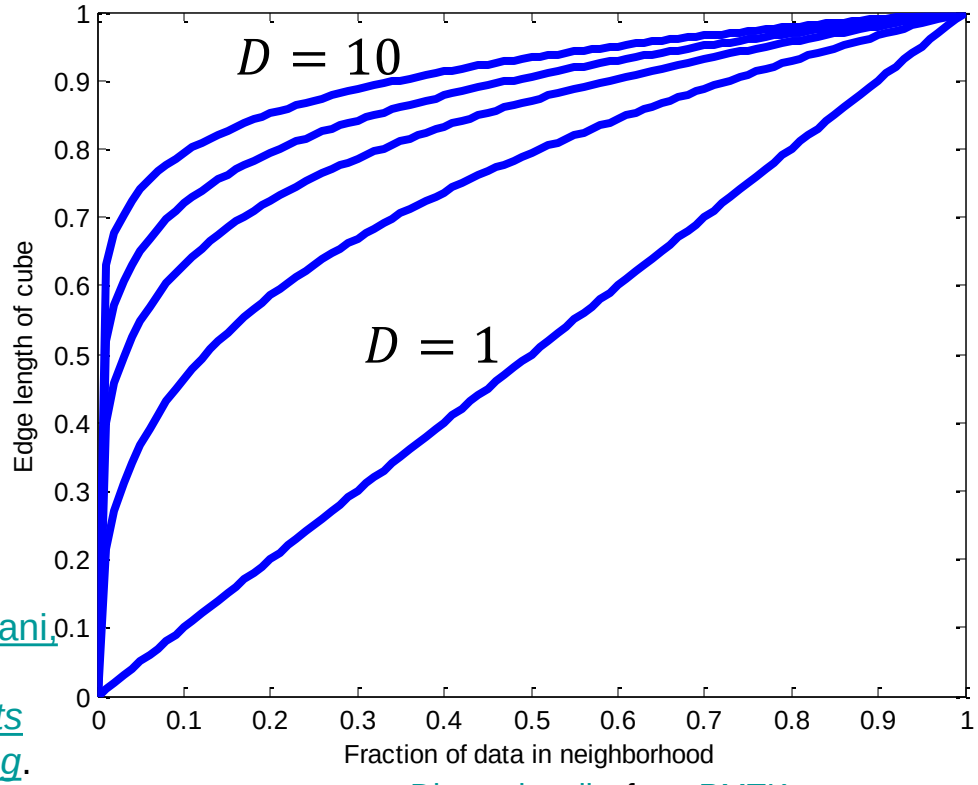
- *K-NN can come within a factor of 2 from the best possible performance as $N \rightarrow \infty$*
- KNN classifiers do not work well with high dimensional inputs.

- Cover, T. and P. Hart (1967). Nearest neighbor pattern classification. *IEEE Trans. Inform. Theory* *13*(1), 21– 27.

Curse of Dimensionality

- Consider *data uniformly distributed in a unit cube in D −dimensions*. We apply K −NN. (a) We embed a small cube of side s inside a larger unit cube. (b) We plot the edge length of a cube needed to cover a desired fraction f of the data points as a function of the number D of dimensions





curseDimensionality from PMTK

$$e_D(f) = f^{1/D}$$

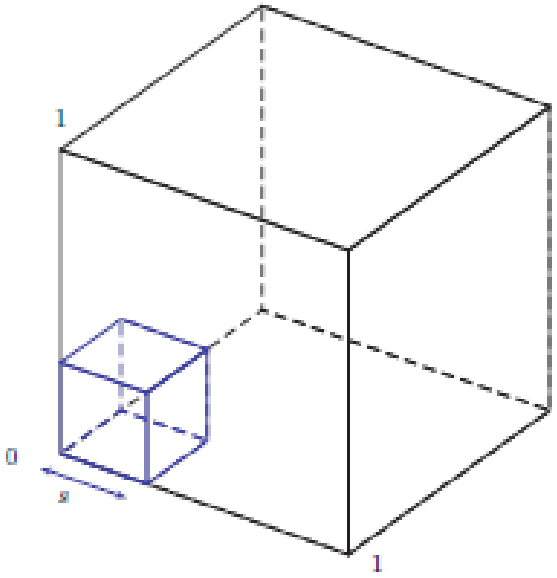
$$e_{10}(0.1) = 0.8$$

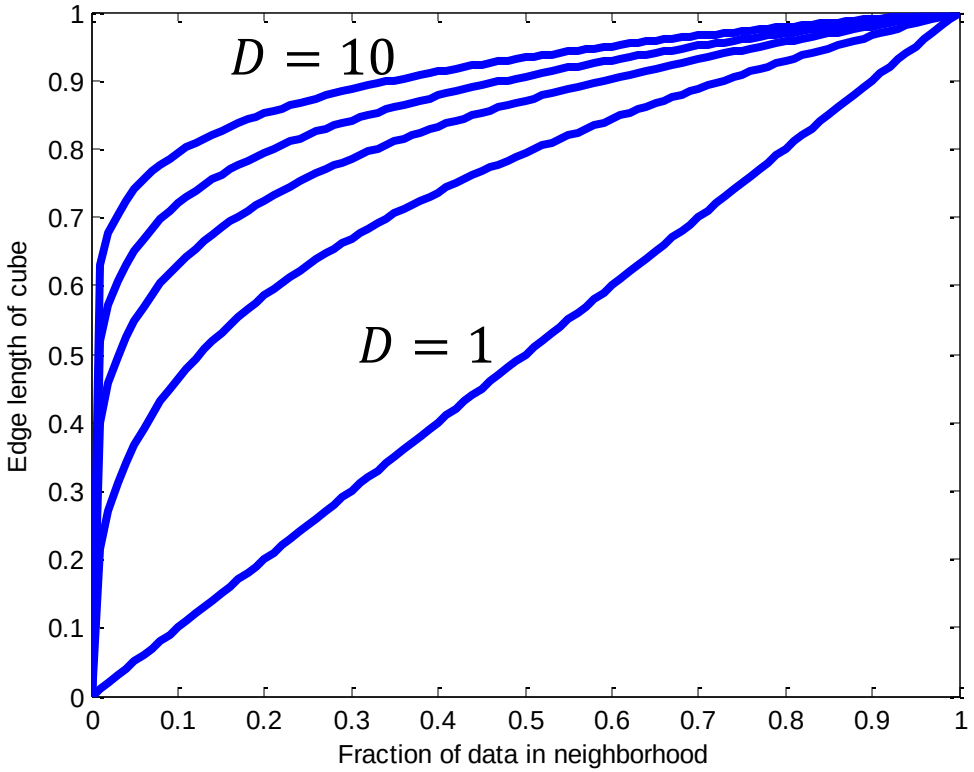
$$e_{10}(0.01) = 0.63$$

- Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning*. Springer. 2nd edition.

Curse of Dimensionality

- KNN is no longer very local in high dimensions.
- Neighbors so far away are not good predictors about the behavior of the input-output function at a given point.





curseDimensionality from PMTK

$$e_D(f) = f^{1/D}$$

$$e_{10}(0.1) = 0.8$$

$$e_{10}(0.01) = 0.63$$

- Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning*. Springer. 2nd edition.

Curse of Dimensionality

- Combat the curse of dimensionality by making assumptions about the nature of the data distribution.

- These assumptions, known as ***inductive bias***, are often embodied in the form of a parametric model (i.e. a model with a fixed number of parameters).

- We review such parametric models next for classification and regression.

Linear Regression

- One of the most widely used models for regression is known as linear regression.

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + \varepsilon = \sum_{i=1}^{D} w_i x_i + \varepsilon$$

- The model weight vector is $\boldsymbol{w}$, and ε is the residual error between our linear predictions and the true response.
- For Gaussian error, we can rewrite the model in the following form:

$$p(y \mid \mathbf{x}, \theta) = \mathcal{N}\left(y \mid \mu(\mathbf{x}), \sigma^2(\mathbf{x})\right)$$

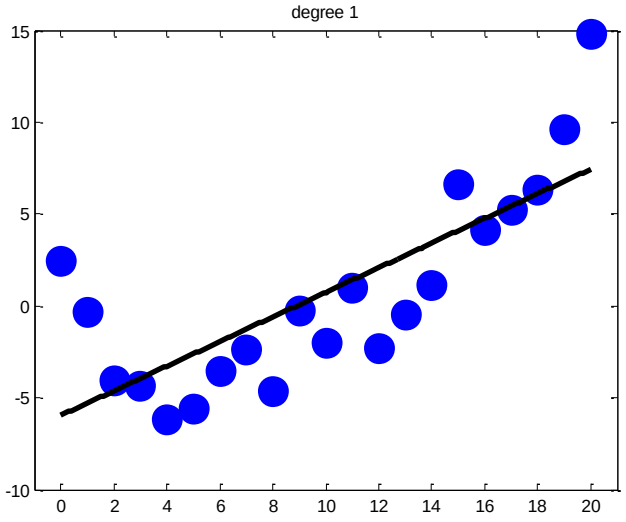
$$Often: \mu(\mathbf{x}) = \mathbf{w}^T \mathbf{x}, \sigma^2(\mathbf{x}) = \sigma^2, \theta = \left(\mathbf{w}, \sigma^2\right)$$

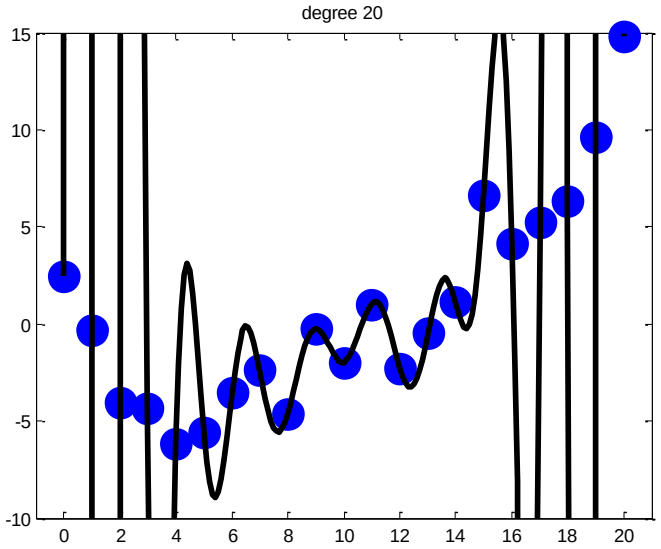
Linear Regression

- Linear regression can be made to model non-linear relationships by replacing $\boldsymbol{x}$ with non-linear basis function of the inputs, $\phi(\boldsymbol{x})$

$$p(y \mid \mathbf{x}, \theta) = \mathcal{N}\left(y \mid \mathbf{w}^T \phi(\mathbf{x}), \sigma^2\right)$$

$$\text{E.g. Polynomial Regression}: \phi(\mathbf{x}) = \left(1, x, x^2, \ldots, x^d\right)$$





linregPolyVsDegree from PMTK

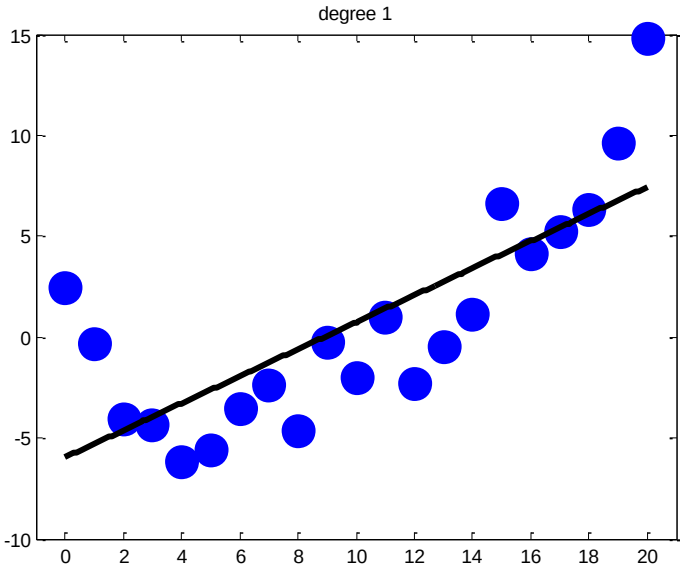
Other choices of $\phi(\boldsymbol{x})$

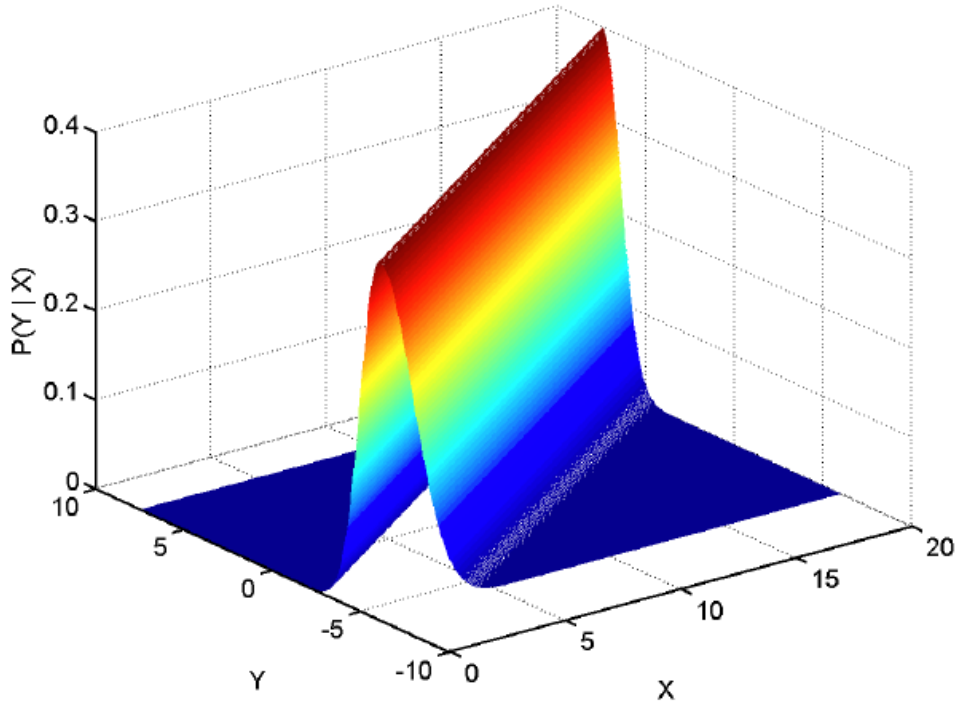
- Support vector machines
- Kernel functions (Gaussians, etc)
- Neural nets
- Etc.

Linear Regression

- A visualization of linear regression is shown here. Note that that the density falls off exponentially from the regression line:

$$p(y \mid x, \theta) = \mathcal{N}\left(y \mid w_0 + w_1 x, \sigma^2\right)$$





linregWedgeDemo2 from PMTK

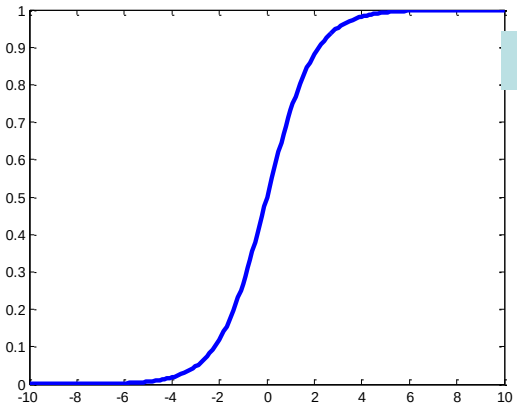
Logistic Regression

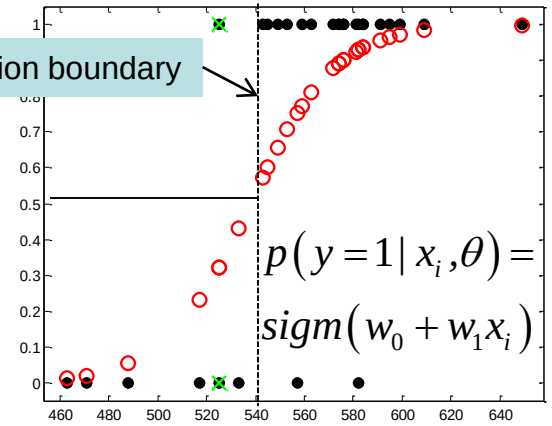
- For a binary response $y \in \{0,1\}$, we replace the Gaussian distribution for y with a *Bernoulli distribution*. That is, we use

$$p(y \mid \mathbf{x}, \theta) = \mathcal{Bern}(y \mid \mu(\mathbf{x}))$$

$$\mu(\mathbf{x}) = sigm(\mathbf{w}^T \mathbf{x}), \; sigm(\eta) = \frac{e^{\eta}}{1+e^{\eta}} = \frac{1}{1+e^{-\eta}}$$

- *The sigmoid function (S –shaped) shown below is also known as the logistic or logit function*.





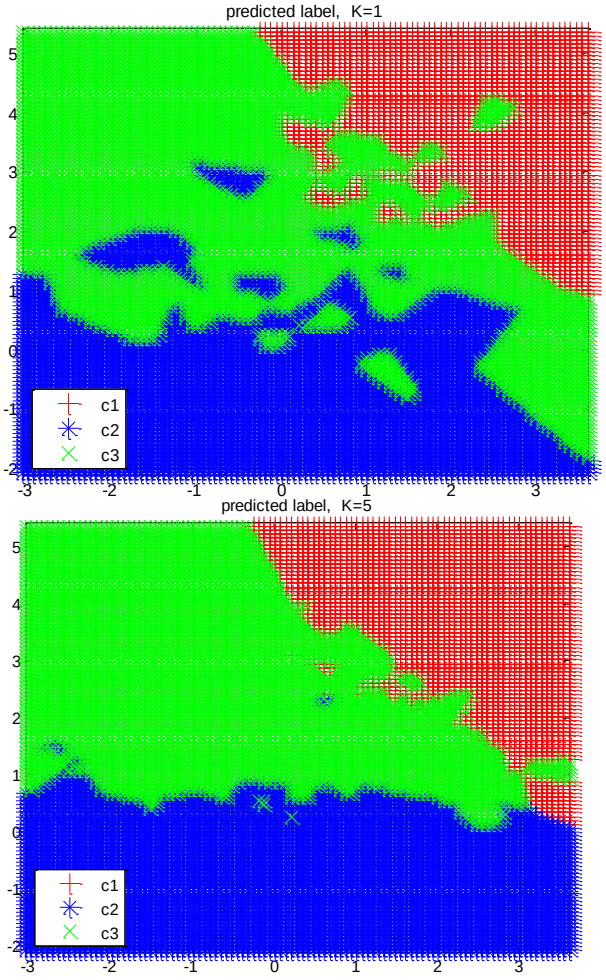
- Logistic regression for SAT scores. Black dots are the data.
- Red circles: predicted probabilities.
- *Non-zero error rate – even on the training data.*
- Green crosses denote 2 students with the same score ($\boldsymbol{x}$) but different training labels (one passed, $y = 1$, one failed, $y = 0$).
- *Data not (linearly) separable. There is no straight line separating the 0's from the 1's.*

sigmoidPlot, logregSATdemo from PMTK

$$\hat{y} = 1 \; iff \; p(y = 1 | x, \hat{w}) > 0.5$$

Overfitting

- Don't model every minor variation in the input, since this is likely noise than true signal.
- In the KNN classifier, the value of K has a large effect on the behavior of this model.
  - When $K = 1$ (complex model), the method makes no errors on the training set (we just return the labels of the original training points), but the resulting prediction is very wiggly.
  - Using $K = 5$ results in a smoother prediction surface, because we are averaging over a larger neighborhood.
  - As K increases (simple model), the predictions become smoother until, in the limit of $K = N$, we end up predicting the majority label of the whole data set.



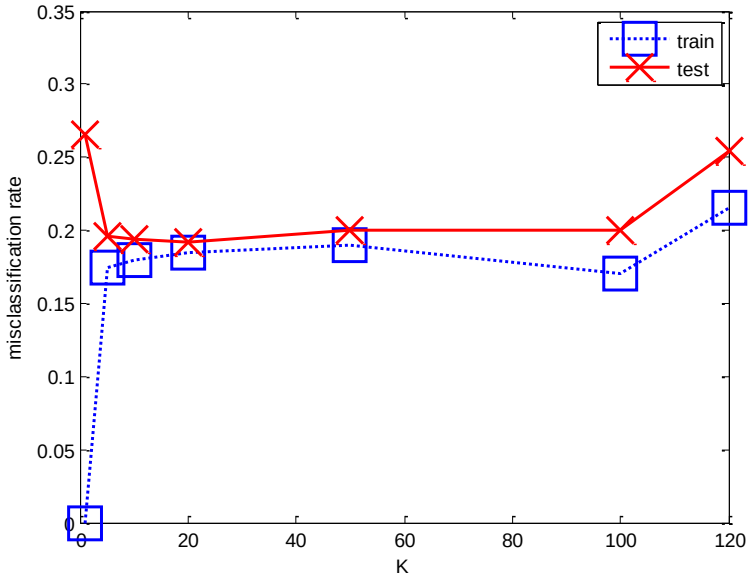
knnClassifyDemo from PMTK

Model Selection

- When we have models of different complexity (regression models with different degree polynomials, KNN classifiers with different K, etc.), how should we pick the right one?
- A natural approach is to *compute the misclassification rate on the training set for each method*. With $f(x)$ our classifier:

$$err(f, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(f(x_i) \neq y_i)$$

knnClassifyDemo from PMTK

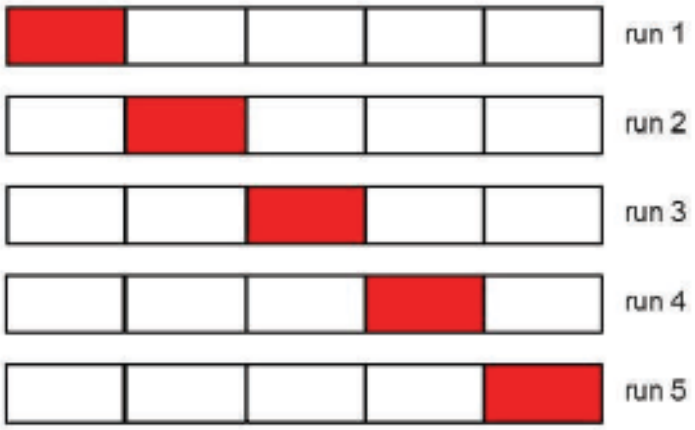


- Misclassification rate vs K in a K −nearest neighbor classifier.
- On the left, K is small, the model is complex and we overfit.
- On the right, K is large, the model is simple and we underfit.
- *Training set* (size 200). *Test set* (size 500) is more appropriate for *generalization*. Here we select K in the range $10..100$.

Cross Validation (CV)

- In cross validation (CV), we split the training data into K folds; then, for each fold $k \in \{1, \ldots, K\}$, we train on all the folds but the k'th, and test on the k'th.

- We then compute the error averaged over all the folds, and use this as a proxy for the test error.



- It is common to use 5-fold CV.

- If we set $K = N$, then we get a method called ***leave-one out cross validation***, *or* ***LOOCV***, since in fold i, we train on all the data cases except for i, and then test on i.

No Free Lunch Theorem

- *There is no universally best model applicable to all problems - this is called the no free lunch theorem*.
- A set of assumptions that works well in one domain may work poorly in another.
- *We need different types of models, to cover the wide variety of data that occur in various application domains.*
- For each model, there are different algorithms to train the model, which make different *speed-accuracy-complexity tradeoffs*.

- Wolpert, D. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation 8*(7), 1341–1390.